SLAM Algorithms

EDAA – G06

João Martins
Henrique Ribeiro – João Costa
Tiago Duarte

Simultaneous Location And Mapping

- **Goal** – map an environment navigated by and autonomous vehicle, while simultaneously locating it in the map;
- **Challenges:**
  - No access to pre-existing maps or external devices;
  - Focus on sub-aquatic SLAM ⇒ difficult access and extra data noise;
- **Datasets** – the group will have access to sonar data measured by CRAS.

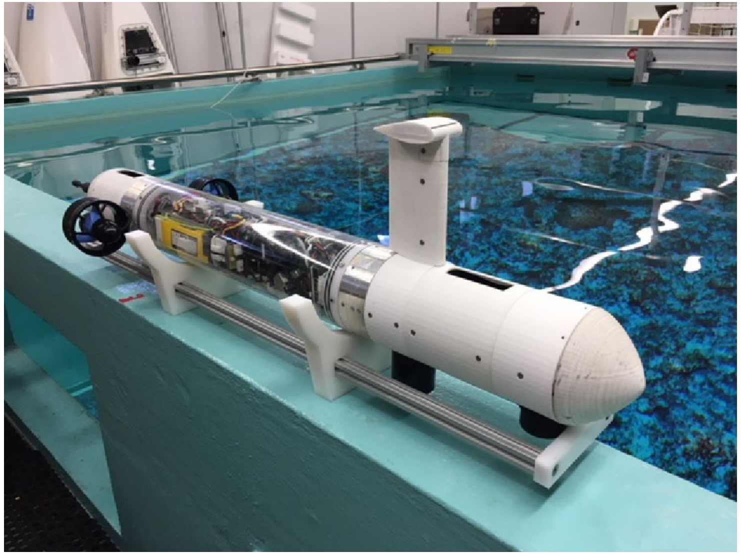

Fig 1. UAV used to collect the datasets.

Roadmap

1. **Probabilistic Mapping Problem**
    1.1. Small recap of sonar data
    1.2. Probabilistic mapping
2. **Edge detection**
    2.1. Simple threshold approach
    2.2. Canny filter
3. **Noise Reduction Methods**
    3.1. Computer vision similarities
    3.2. Kalman filter
4. **Raycast Algorithm**
    4.1. Bresenham's line Algorithm

Sonar data

- A sonar mounted on a vehicle collects environment data;
- Sonar data contains noise and other undesirable effects (e.g. multipath);
- **Fig 2.** Illustrates the raw data of the sonar and the problems present in this data:
    - Reflections from the body of the vehicle (self reflections);
    - Multipath effects when signals go through the tank walls;
    - Noise affecting detection of features (e.g. tank walls and floater).
- It is important to clean this data and find the distance to the first feature for each measurement.

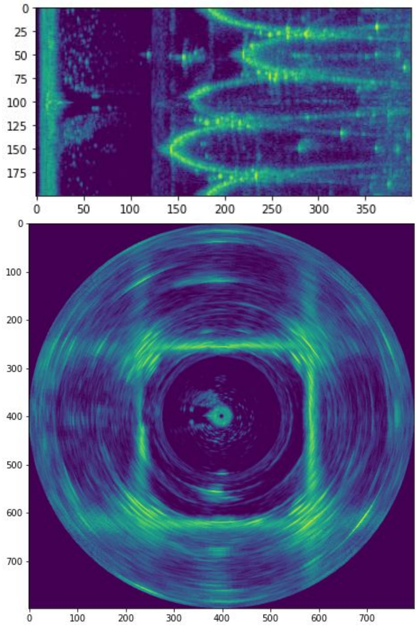

Fig 2. Dataset representation in polar and Cartesian coordinates.

Sonar Data

- Sonar rotates around itself
- Sends/measures waves in a cone
  - But we will only consider the 2D problem in this first part
- Each beam has multiple intensities across several intervals

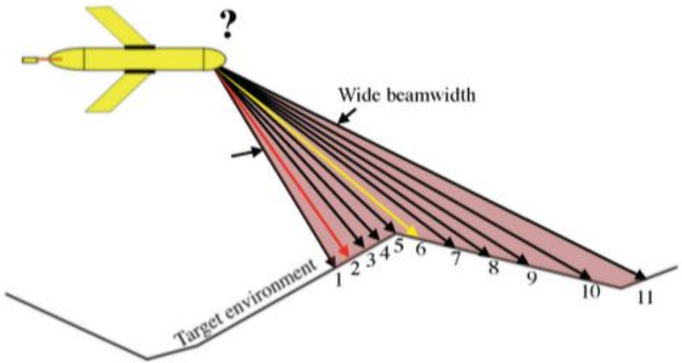

Fig 3. Sonar beam representation 2d [PAPER]

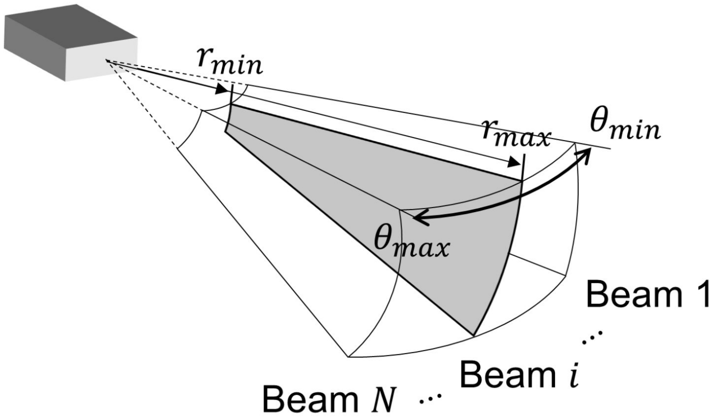

Fig 4. Sonar beam representation 3d [PAPER]

Map updates

- Find the first obstacle that a ray intersects
- Update values that we know that aren’t occupied
- **How do we update the probabilities?**
- **How do we find cells that the beam intersects?**
- **How do we detect that we’ve hit an obstacle?**

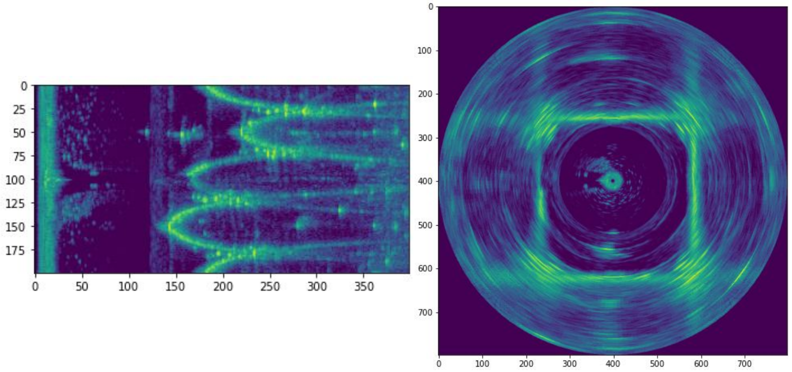

Fig 5. Sonar data in polar and Cartesian coordinates

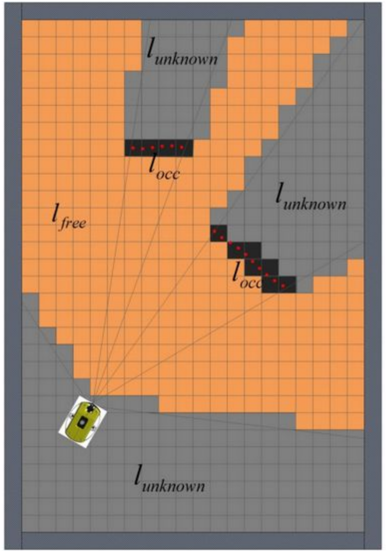

Fig 6. Beam cast representation[1]

Probabilistic Mapping

Probabilistic Mapping [2]

- Use conditional probabilities to update map

$$P(n|z_{1:t})$$

Fig 7. Probability that cell is occupied given all measurements

$$P(n|z_t)$$

Fig 8. Probability that cell is occupied using last measurement

$$P(n|z_{1:t-1})$$

Fig 9. Probability that cell is occupied from past measurements

- Using Bayes theorem, we can deduce:

$$P(n|z_{1:t}) = \left[1 + \frac{1 - P(n|z_t)}{P(n|z_t)} \frac{1 - P(n|z_{1:t-1})}{P(n|z_{1:t-1})} \frac{P(n)}{1 - P(n)}\right]^{-1}$$

Fig 10. Probability update formula

Probabilistic Mapping [2]

$$P(n|z_{1:t}) = \left[1 + \frac{1-P(n|z_t)}{P(n|z_t)} \frac{1-P(n|z_{1:t-1})}{P(n|z_{1:t-1})} \frac{P(n)}{1-P(n)}\right]^{-1}$$

Fig 11. Probability update formula

- Which can be converted to log-odds notation:
  - More efficient – Reduces multiplications/fractions

$$L(n|z_{1:t}) = L(n|z_{1:t-1}) + L(n|z_t)$$

Fig 12. Probability update formula in log odds

$$L(n) = log(\frac{P(n)}{1-P(n)})$$

Fig 13. Log odds formula

Map updates

- Find the first obstacle that a ray intersects
- Update values that we know that aren't occupied
- How do we update the probabilities?
- **How do we find cells that the beam intersects?**
- **How do we detect that we've hit an obstacle?**

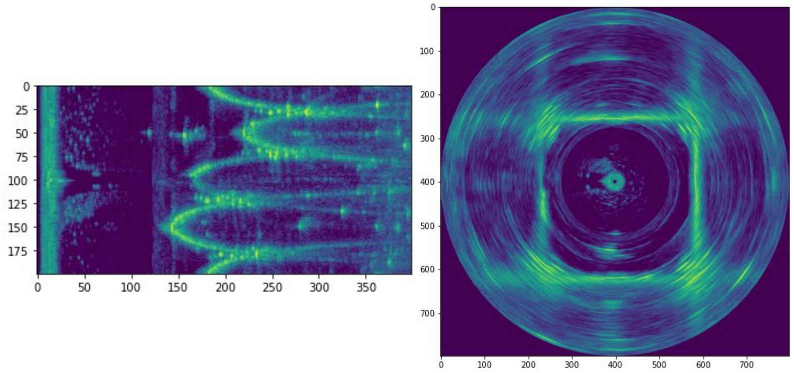

Fig 5. Sonar data in polar and Cartesian coordinates

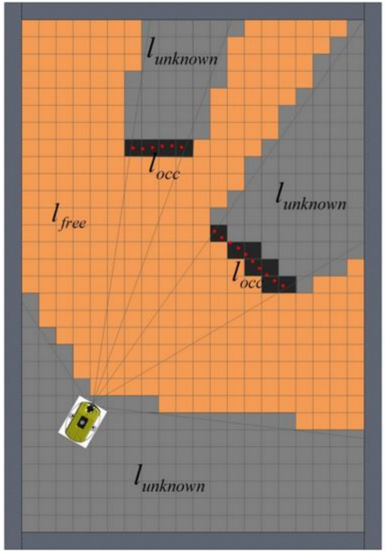

Fig 6. Beam cast representation [1]

Raycasting Algorithms

Raycasting

- Given two points that form a line:
  - Find which cells map to it
- Update cell probabilities

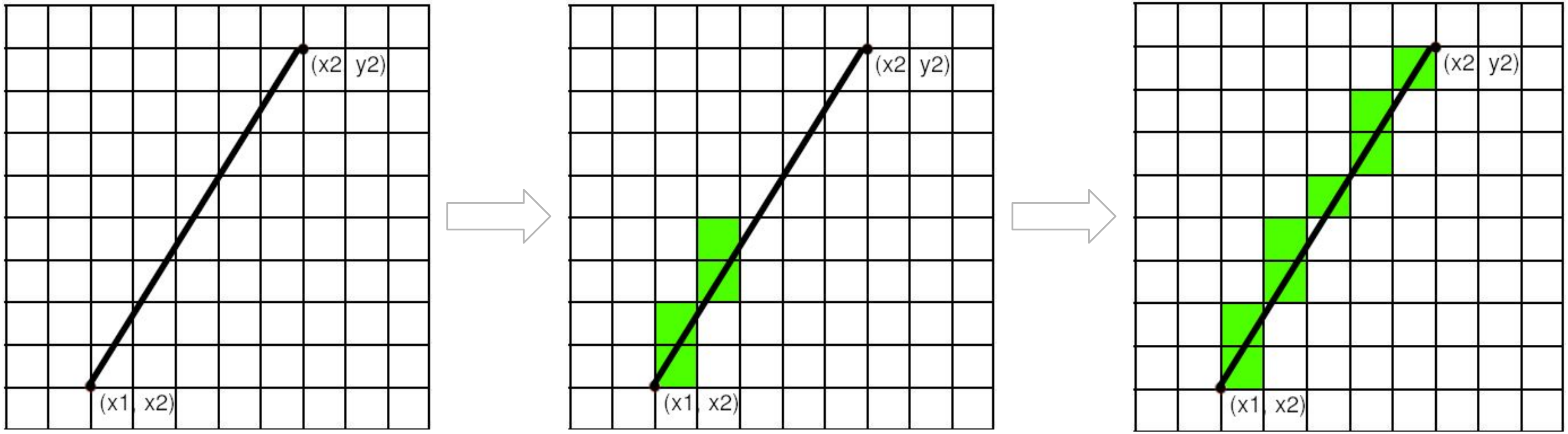

Fig 14. Raycasting visualized

Raycasting

- Detect an intersection with an object → Edge detection
- Update cell probability
  - Cell is before object → free
  - Cell holds object → occupied
  - Cell is after object → unknown

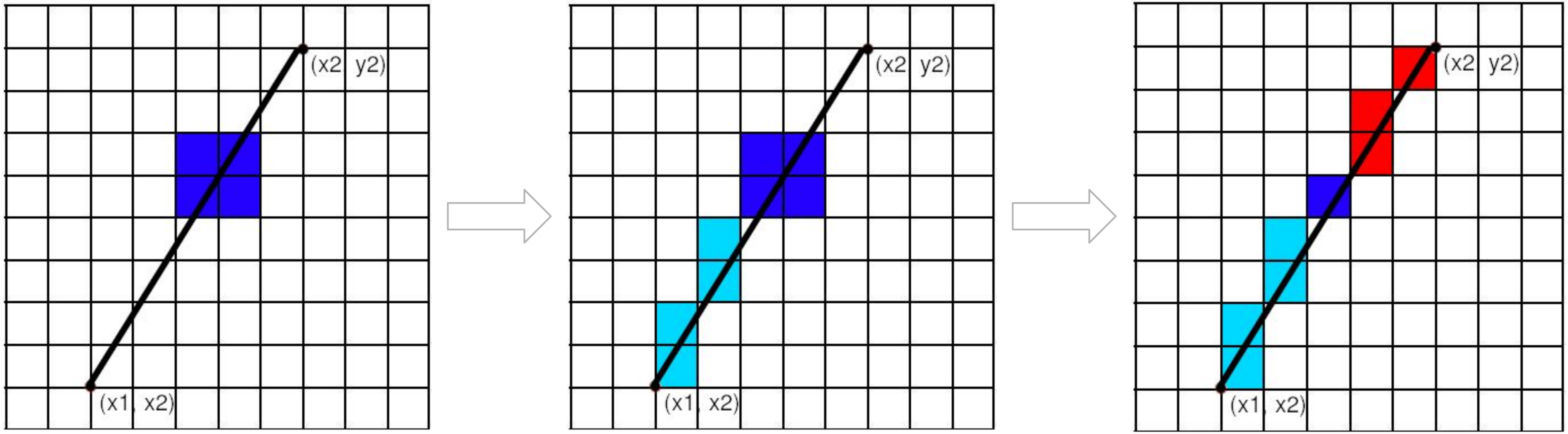

Fig 15. Using raycasts to identify cells behind/in front of obstacle

Bresenham’s Line Algorithm

- Select closest cell to the line based on error
  - Error is calculated with dx and dy values

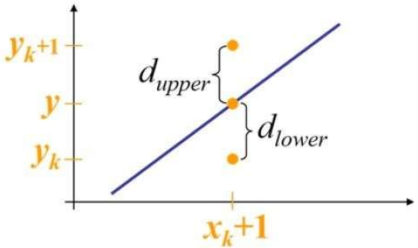

Fig 17. Cell distances from decision point

```
def bresenham(start, end):
    (x0, y0) = start
    (x1, y1) = end

    dx, dy = abs(x1 - x0), abs(y1 - y0)
    x, y = x0, y0
    cells = []
    p = 2*dx - dy
    while (x <= x1):
        cells.append((x, y))
        x += 1
        if p < 0:
            p += 2 * dy
        else:
            p += 2*dy - 2*dx
            y += 1
    return cells

```

Fig 16. Bresenham Algorithm

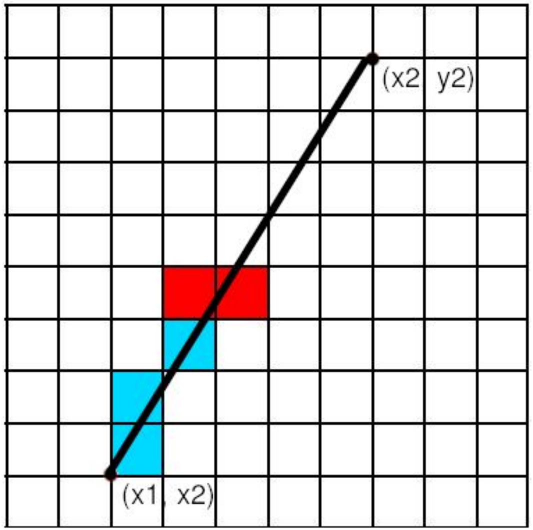

Fig 18. Bresenham’s line Algorithm - Decision visualized

Map updates

- Find the first obstacle that a ray intersects
- Update values that we know that aren’t occupied
- How do we update the probabilities?
- How do we find cells that the beam intersects?
- **How do we detect that we’ve hit an obstacle?**

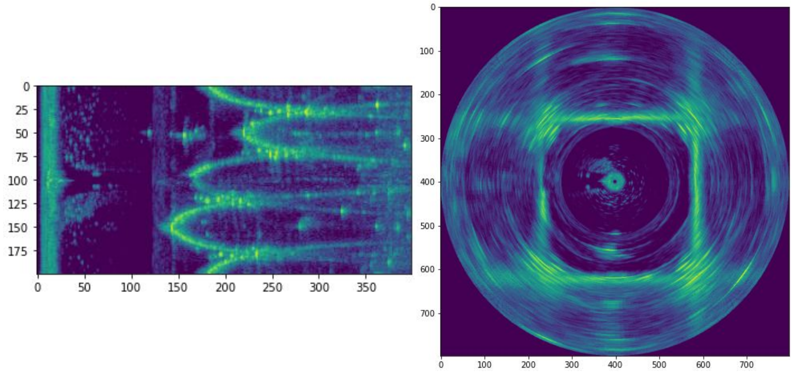

Fig 5. Sonar data in polar and Cartesian coordinates

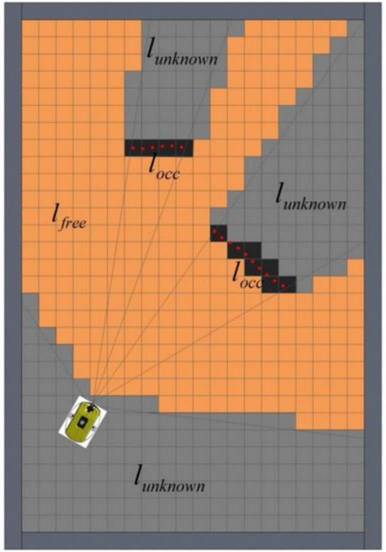

Fig 6. Beam cast representation [1]

Edge Detection

Canny Filter [3]

- Uses two thresholds and edge tracking by hysteresis
    - Necessary to find the best combination of thresholds
- Increased blur provides best results

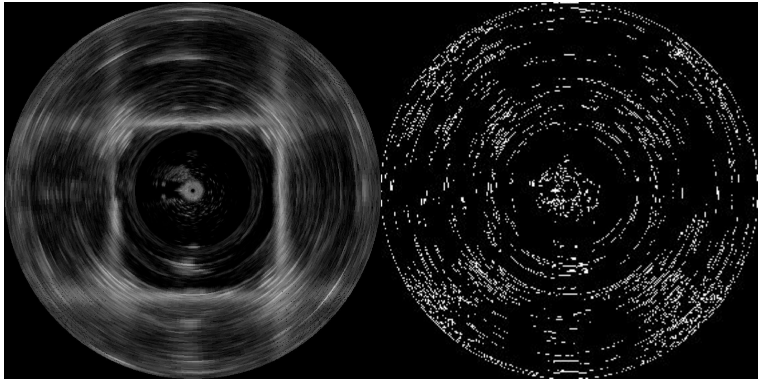

Fig 19. Original scan.

Fig 20. Original scan with canny filter.

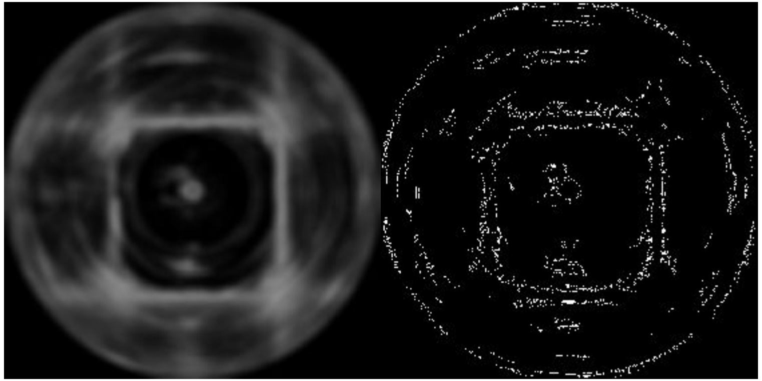

Fig 21. Blurred scan.

Fig 22. Blurred scan with canny filter.

Simple threshold approach [2]

- Calculate variation of intensities
- Define a threshold
    - Variations above that threshold approach
- Increased blur provides best results

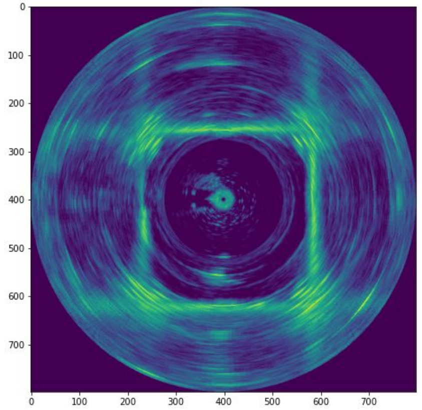

Fig 23. Original Image

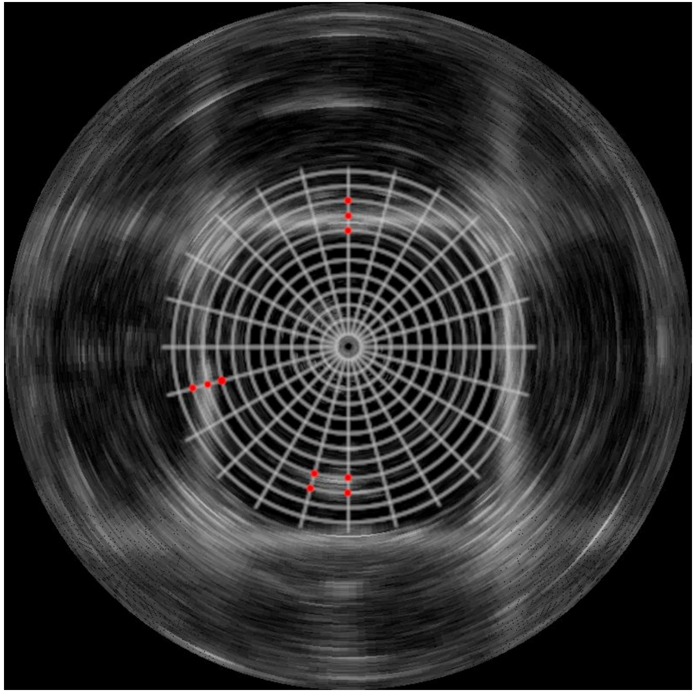

Fig 24. Scan with identified edges using simple threshold approach

Canny Filter

Produces better results 01

Harder to tune 02

Simple Threshold

Easier to implement 01

Sensitive to noise 02

Less flexible 03

Noise Reduction Methods

Computer Vision Similarities [4][5]

- Consider sonar data as a one channel image that has noise
- Treat noise with computer vision algorithms

Fig 25. Image with and without noise

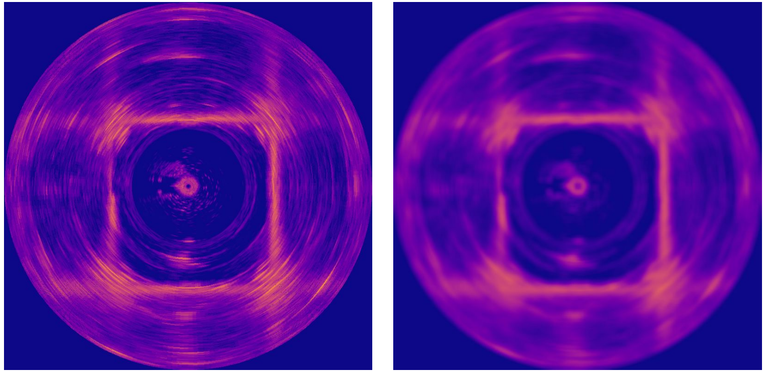

Fig 26. Sonar data with noise and with reduced noise

C.V. Smoothing Filters

- Use blurring to reduce noise
- Improves drastically edge detection

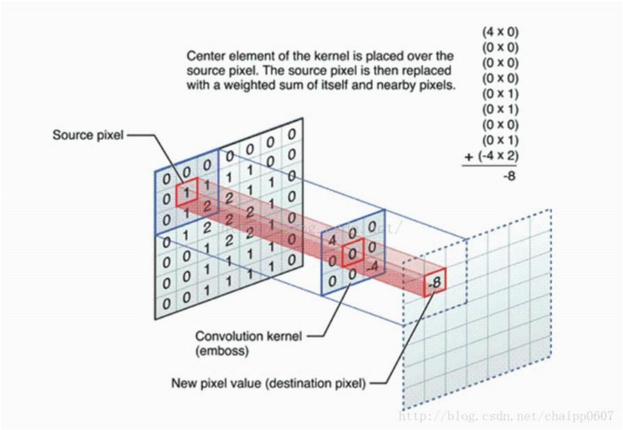

Fig 31. Convolution visualized

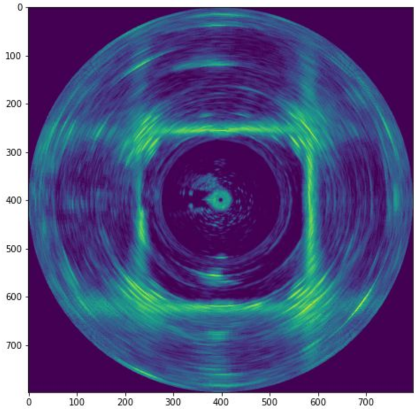

Fig 27. Dataset representation in Cartesian coordinates.

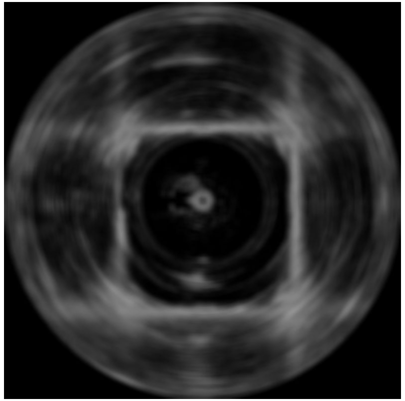

Fig 28. Gaussian filter

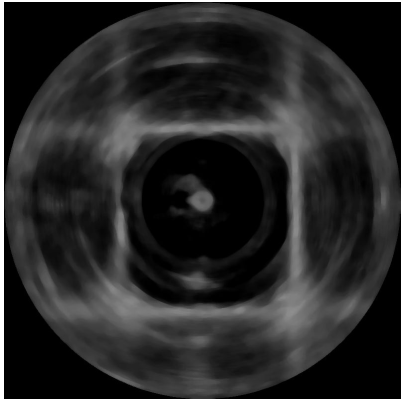

Fig 29. Median filter

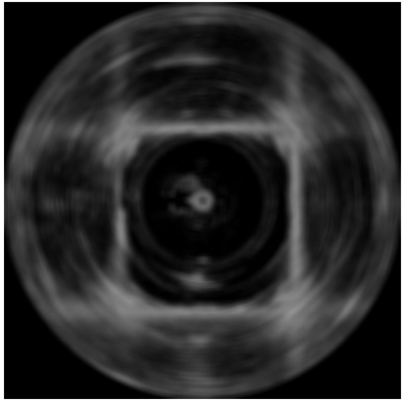

Fig 30. Mean filter

C.V. Smoothing Filters

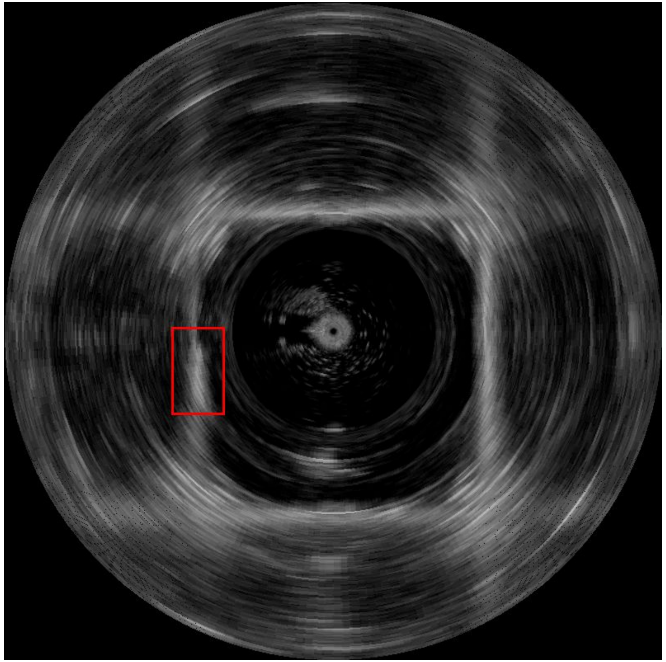

Fig 32. Original scan image

Fig 33. Original selection

Fig 34. Gaussian

Fig 35. Median

Fig 36. Mean

Extended Kalman Filter

- Define the system state - A full sonar scan
- Define model - function describing sonar/environment movement
- Derive new state from measurements and estimation
    - Measurement - New sonar scan (new state)
    - Estimate - From function and old state

- Estimate new state from measurements and previous state

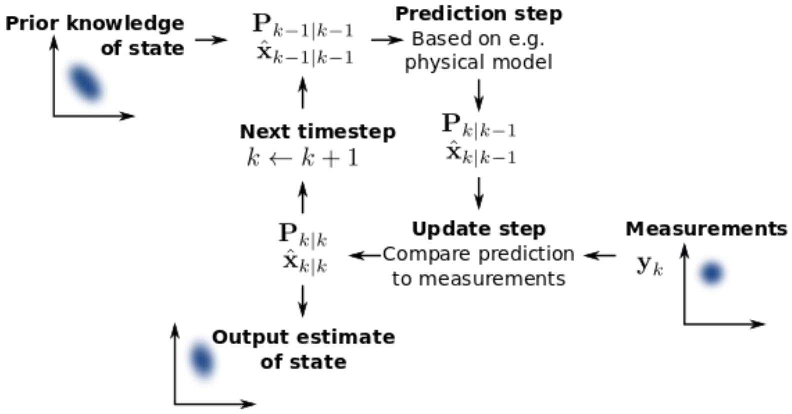

Fig 37. Kalman filter algorithm visualize [6]

Extended Kalman Filter - Problems

- Need to derive model from a set of feature points
    - Need to extract these points from data and identify th properly
        - ■ Complex problem - Requires line and corner detection
        - ■ Part of the localization part
        - ■ Out of scope for this project
- Movement function must be linear
    - Requires Taylor series to linearize the model

Model

$$\dot{\mathbf{x}}(t) = f\big(\mathbf{x}(t), \mathbf{u}(t)\big) + \mathbf{w}(t) \qquad \mathbf{w}(t) \sim \mathcal{N}\big(\mathbf{0}, \mathbf{Q}(t)\big)$$
$$\mathbf{z}(t) = h\big(\mathbf{x}(t)\big) + \mathbf{v}(t) \qquad \mathbf{v}(t) \sim \mathcal{N}\big(\mathbf{0}, \mathbf{R}(t)\big)$$

Initialize

$$\hat{\mathbf{x}}(t_0) = E\big[\mathbf{x}(t_0)\big], \mathbf{P}(t_0) = Var\big[\mathbf{x}(t_0)\big]$$

Predict-Update

$$\dot{\hat{\mathbf{x}}}(t) = f\big(\hat{\mathbf{x}}(t), \mathbf{u}(t)\big) + \mathbf{K}(t)\Big(\mathbf{z}(t) - h\big(\hat{\mathbf{x}}(t)\big)\Big)$$
$$\dot{\mathbf{P}}(t) = \mathbf{F}(t)\mathbf{P}(t) + \mathbf{P}(t)\mathbf{F}(t)^{\top} - \mathbf{K}(t)\mathbf{H}(t)\mathbf{P}(t) + \mathbf{Q}(t)$$
$$\mathbf{K}(t) = \mathbf{P}(t)\mathbf{H}(t)^{\top}\mathbf{R}(t)^{-1}$$
$$\mathbf{F}(t) = \left.\frac{\partial f}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}(t), \mathbf{u}(t)}$$
$$\mathbf{H}(t) = \left.\frac{\partial h}{\partial \mathbf{x}}\right|_{\hat{\mathbf{x}}(t)}$$

Fig 38. Kalman filter steps described [6]

E.K.F.

Much more accurate 01

Requires feature point extraction 02

Gaussian Filter

Easier to implement 01

Doesn’t take movement into account 02

Efficiency Challenges

- Algorithms need to run under some constraints
  - 40 ms per beam
  - 200 beams per scan

- Noise reduction algorithms need a full scan to work
  - For instance, smoothing filters and E.K.F.
  - Possible efficiency/accuracy tradeoffs

References

[1] - Real Time Obstacle Detection in a Water Tank Environment and its Experimental Study - Ji-Hong Li, Mun-Jik Lee, Won-Seok Lee, Jung-Tae Kim, Hyung-Joo Kang, Jin-Ho Suh

[2] - Underwater mapping using a SONAR - João Fula

[3] - Bottom Tracking Method Based on LOG/Canny and the Threshold Method for Side-scan Sonar - Shengping Wang, Hongtao Li, Xiaoyu Li, Jiansong Yang and Quanhong Feng

[4] - Automatic target detection of sonar images using multi-modal threshold and connected component theory - Subhra Kanti Das, Soma Banerjee, Dibyendu Pal, Sambhunath Nandy, Sankar Nath Shome & Somnath Mukherjee

[5] - Techniques adopted in the post processing of active sonar data from Royapuram site-off Chennai - Mahimol Eldhose, Dhilsha Rajapan, Shijo Zacharia, D.S. Sreedev & M. A. Atmanand

[6] - Kalman Filter - Wikipedia

Questions?